

.....

# **Dynamic Appearance Model v.0.1**

**Project Documentation**



# Table of Contents

---

<b>1 Overview</b>	
1.1 News and Status .....	1
1.2 Technical Documents .....	2
1.2.1 Transformation Requirements .....	3
1.2.2 DAM Model Proposal .....	9
1.2.3 DAM Test Cases .....	16
1.2.4 Extending SCORM .....	23
1.2.5 DAM: Flow of Information .....	29
<b>2 Miscellaneous</b>	
2.1 DAM Implementation Notes .....	41



## 1.1 News and Status

---

### News and Status

#### **31 May 2004 - First draft of Extensions to SCORM**

- First draft of Extensions to SCORM document.
- Working on DocBook transformation implementation.

#### **02 April 2004 - First draft of Technical Documents**

- First draft of DAM Model Proposal.
- First draft of DAM: Flow of Information.
- DocBook to XHTML test servlet committed.

#### **12 March 2004 - First commit of ADL Sample RTE Linux port**

- Replaced ODBC MS Access with JDBC Postgres.
- Implemented connection pooling.
- Changed paths to be platform independant.

Tested and working with Photoshop\_Compency.zip.

## 1.2 Technical Documents

---

### DAM Technical Documents

#### Overview of the DAM Technical Documents

Document	Description
<a href="#">DAM Transformation Requirements</a>	The implementation of the DAM will consist of a series of supported transformations originally defined in the <a href="#">statement of work</a> . This document goes over in further detail the requirements of the content and user agent in accomplishing these transformations.
<a href="#">DAM Model Proposal</a>	Before development can begin we need to agree on a model implementation of the DAM. This document is the first proposal addressing this action item.
<a href="#">DAM: Flow of Information</a>	This document diagrams each transformation in terms of its inputs and outputs.
<a href="#">DAM Test Cases</a>	SCO requirements for testing the transformations.
<a href="#">Extending SCORM</a>	Proposed extensions and bindings to SCORM.
<a href="#">DAM Implementation Notes</a>	Developer notes, reminders and musings.

## 1.2.1 Transformation Requirements

---

### DAM Transformation Requirements

#### Acronyms

- ACCLIP - [Accessibility for Learner Information Package](#)
- DAM - Dynamic Appearance Model
- SCO - Sharable Content Object
- SOW - Statement of Work
- TILE - [The Inclusive Learning Exchange](#)
- WCAG - [Web Content Accessibility Guidelines](#)
- W3C - [World Wide Web Consortium](#)

#### Introduction

This document outlines the requirements for a SCO to allow the transformations outlined in the DAM/SCORM/SOW. The SOW has a table of transformations with a key indicating whether that transformation is applicable to branding. This document lists those transformations that were declared as relevant to branding (i.e., the "maybe's" have been dropped). For each transformation, there is a brief description of how TILE handles the transformation, and, based on that, what requirements there would be for the SCO in order that we can co-opt the TILE technique.

#### Text Rendering

This category is meant to group together transformations involving:

- font family
- text size,
- text colour,
- background colour,
- and highlight colour.

TILE handles this transformation using CSS, implying that the SCO must be transformable via CSS. CSS works with only XML and HTML documents; hence, the SCO must be XML or HTML. Since it is much easier to style HTML with CSS, and since the point of this project is to demonstrate transformations, we recommend that the SCO be HTML.

Also, it's likely that DAM will use XSL for other transformations. XSL cannot be used with straight

HTML (W3C). We therefore recommend that the SCO be XHTML.

At the February face-to-face meeting, Peter proposed that all SCOs be in a neutral data format, from which various specific formats could be generated. Examples of specific formats include PDF and HTML. The suggested neutral data format was DocBook. With respect to a text rendering transformation, DocBook is fine with the proviso that the transformation will be applied to the XHTML version of the SCO generated from DocBook.

### Layout

TILE currently does not handle this; neither is there anything in the acclip to define layout. It is tempting to use CSS for this as well, but even for simple layouts, a CSS can be daunting. The use of xhtml <table> elements is easier for more complex layouts. Also, the support for CSS layout techniques varies between many mainstream browsers and is sometimes not supported at all; whereas table support is fairly universal.

In terms of branding, one possibility is that each brand pre-defines its own layout, in CSS. In that case, the CSS is "canned," and the transformation amounts to choosing the correct institutional branding layout to use with the SCO at runtime. A branded layout CSS implementation has the prerequisite that the SCO be in a common data format with standard CSS 'class' and 'id' attributes defined. This is typically handled using <div> and <table> elements. That is, the major areas of the document are declared using <div> and then are positioned by putting those <div> elements into the correct cells of a table.

In short, the SCO is either transformed via CSS or by transforming a <table>, or both. This again requires that the SCO be XHTML. It also needs its major sections identified clearly using class and id attributes, and possibly <div> elements.

If a neutral data format for a DAM-SCO is decided upon, this format could support (explicitly or implicitly) a set of various layout constructs in which a layout-XSLT transformation (and optionally afterwards, a CSS) could be applied. For example, the data format could support such layout constructs as: header, footer, menubar, navigation bar, logo, sections, subsections, etc. in which a 'branding' layout-XSLT transformation could be generated and applied to each DAM-SCO for the given institution at runtime. Some potential data formats to investigate further would be: XDoc (<http://maven.apache.org>), StyleBook, and DocBook.

### Document (DOM) order

TILE does not handle this transformation. There are two ways to reorder the DOM, either using code (DOM api manipulations, cf. Xalan), or via XSLT. The requirement again is that the SCO be XHTML. Also, the sections to be reordered must be clearly identified in the markup.

This transformation is similar to layout above and, as a result, has the same common data format prerequisites. It is a technically different transformation from layout -- layout does *not* modify the order of the elements in the document. Layout affects where on the page the elements are positioned visually. A document order transformation does change the order of the in the DOM tree, but without regard to where those elements are positioned visually. The visual result of both transformations may, however, be very similar.

In any case, this transformation entails the restrictions that a SCO have a DOM representation, and that

the parts to reorganize are clearly marked.

### **Exclusion (Navigation)**

In TILE, this refers to a table of contents that lists topics and allows a user to choose among. Exclusion is the availability (and visibility) of those topics. It is important to note that it is at the user's request whether topics are excluded.

The table of contents is generated from an item hierarchy in the content package manifest. The excluded topics are based on user preferences handled by settings outside of the acclip. This results ultimately in the availability of some resource or content; in SCORM-speak, the availability of a SCO. Thus, an excluded topic is an excluded SCO.

With respect to branding, this amounts to one brand excluding a set of SCOs referenced from a content package, while another brand excludes another set of SCOs. The transformation starts with an item hierarchy and marks items as excluded or included. In short, there are no requirements vis-À-vis the SCO, rather the requirements pertain to the SCO's metadata. The SCO must be identifiable as one to exclude, and that information resides outside of the SCO.

### **Alternate Content**

With respect to the acclip and TILE, there are a number of alternate content transformations. Each of these is dealt with in a sub-section. The transformations are:

- [content density](#)
- [content views](#)
- [alternatives to visual](#)
- [alternatives to audio](#)
- [alternatives to text](#)

### **Content Density**

Content density is how much each of section of a SCO is visible, where users can toggle between "expanded" or "collapsed." This is implemented in TILE using a JavaScript function. The requirement for the SCO is that potentially 'dense' sections of its content be identifiable, and that those sections can be collapsed/expanded via a JavaScript function. That is, the 'dense' section must be a child (or leaf) block-level section.

In terms of branding, this transformation would give an institution a particular 'feel' in displaying large portions of content that have the potential of being less-densely represented. For example, if an institution implemented search functionality across all the SCO's they support, the display of the description of SCO's which matched the given search constraint could be toggled as follows based on the branding content density property:

Collapsed:

The Reading & Note-Taking: Textbooks course is comprised of three parts: Reading, Taking Notes, and Writing Summaries. These three components will together enable you to understand the process of reading and taking notes. They will help you learn about the relation to good study habits, reading

practices and writing assignments. It is recommend ... [Read More (350 of 553 Characters)]

Expanded:

The Reading & Note-Taking: Textbooks course is comprised of three parts: Reading, Taking Notes, and Writing Summaries. These three components will together enable you to understand the process of reading and taking notes. They will help you learn about the relation to good study habits, reading practices and writing assignments. It is recommended that you have completed the Introduction to Online Collaborative Learning and Time Management courses prior to undertaking this one. If you have not taken these, permission from the instructor should be secured prior to beginning this lesson. You might also want to see the Memory & Concentration course. [Read Less]

### Content Views

The acclip allows two values here, namely image intensive, or text intensive. An image intensive learning object is one that communicates its message using lots of graphs, pie charts, images, and so on. A text intensive one uses text, for the most part, to communicate the same message. The transformation amounts to substituting one for the other.

TILE supports a general one-way image intensive to text intensive transformation by utilizing HTML's <img> attributes, "alt" and "longdesc." If requested by the user, TILE substitutes an <img> with a <p> containing the image's alternative and long description text. This transformation requires that the <img> provides proper alternative text descriptions as defined in the WCAG 1.0 guidelines of the W3C. Further investigation will be done on how far this transformation relates to branding and whether or not WCAG 1.0 conformance is a requirement of a DAM-SCO.

The parts of the SCO that are image intensive, for which text is going to substitute, must be clearly marked, and must have "alt" and "longdesc" attributes.

### Alternatives to X

Side bar: acclip alternative content preferences, and what they mean:

- alternative to visual
  - audio alternatives to visual material, e.g. audio (spoken) descriptions of the material.
  - textual alternatives to graphics, e.g., alt text for images.
  - colour avoidance, e.g. for colour blind users.
- alternatives to audio
  - text captions for audio sound track.
  - sign language for audio.
- alternatives to text
  - graphic alternative for text, e.g. stop sign icon instead of the text "Stop."
  - sign language alternative for text.

In short, the alternative content in the acclip involves substituting one mode of communication for another. The acclip *does not* handle substitution of content *within* a modality. With respect to branding, a standard example of an alternate content transformation is substituting one logo for another. That is a

within modality transformation, and the acclip does not deal with this (hence, neither does TILE). This might indicate a deficiency in the acclip, or that the trigger for intra-modality transformations lies outside the acclip.

The requirement for the SCO is that the content to be substituted be clearly identified, via id attributes, and/or <div> elements, so that the "alternative content" of those sections can be swapped in and out.

### **Role of Metadata in Alternate Content Transformation**

Finally, a word about metadata and its relationship to alternate content transformations. First, the content package resource section allows for defining variant resources. This is a TILE extension to the IMS content package schema. Each variant resource is associated with a single resource and references another learning object. The intended semantics are that one learning object is a variation of another. The kind of variation is defined by the variant's meta-data, specifically its /lom/relation information. As with content package resource variants, TILE uses an extended vocabulary for the /lom/relation/kind meta-data. (In fact, the xml machinery used in TILE to make this work is more complex, but this description is enough to communicate the jist). Thus, given that a resource (SCO) is to be launched, and the user has requested, say, an audio alternative, then TILE looks for a variant of that resource whose meta-data states that it is an audio alternative to the main resource.

I see no reason why a similar sort of technique could not be used for alternate content substitutions for branding. The manifest/resource specifies brand variants. The kind of brand is given in that variant's meta-data (although we might have to modify the vocabulary to /lom/relation/kind for this). *Something* tells the system to use brand X; hence, the transformer looks for brand X variants when grabbing resources from the manifest. The issue is: what is the *something*? Acclip? - if so, it's an extended acclip.

The requirement for SCOs is that there be variant SCOs, that their meta-data properly describes the relationship among the variants, and that the content packages be constructed to list the variants.

### **Content Supplements**

Content supplements are similar to alternative content but here, the supplementary material cannot completely replace the original content. An example is text captions -- by themselves they are only a textual transcript of the speech on the audio track. They cannot, by themselves, replace the entire video.

TILE handles supplements in a similar fashion to alternative content, but does require some extra machinery to do something with the supplements. Keeping with the text caption example, given that captions are available for a video, TILE uses SMIL to dynamically combine the original video with the caption text, and then then sends the SMIL markup to the client instead of the original video. A caveat, however: this work is still under development with respect to TILE.

### **Structure Presentation (Navigation)**

The acclip has a user preference for showing or hiding a table of contents. TILE views one purpose of the table of contents as allowing navigation throught the material. The table of contents both shows where the user is in terms of which SCO they are currently viewing, as well as allowing them to "jump" among SCO's rather than be limited by "next" and "previous" links. The table of contents is built from the item hierarchy in the content package organization.

With respect to branding, one brand may rely heavily on a "navigation bar" as part of their look and feel, while another brand finds this less important.

There are no requirements on the SCO for this kind of transformation. The table of contents is derived from the content package, and its visibility from the acclip.

### **Structure Navigation**

This transformation is based on a depth-first vs. breadth-first traversal preference in the acclip. This determines the destination of "next" and "previous" links. Given an item from the content package, and a bread-first traversal, "next" means going from sibling to sibling in the item tree before descending to the children of an item. Depth-first uses the same item tree, but traverses all of an item's descendents first before moving to that item's sibling.

Since this is again determined solely by the acclip and a table of contents, there is no requirement on the SCO themselves.

## 1.2.2 DAM Model Proposal

---

### Use Cases

#### Use Case DAM1: Institutional Branding

**Primary Actor:** Institution

**Stakeholders and Interests:**

- Institution: Wants all content served by its LMS to be styled either by the appearance profile defined in the Content Package of the content or according to its own unique brand or marque depending on whether or not it wishes to override the appearance profile of the Content Package.
- User:
  - Wants to receive requested content in a fast, easy manner.
  - Wants to receive information according to their preferences (colours, sizes, etc.)

**Preconditions:** Content Package defines and bundles an appearance profile. Institution has an appearance profile.

**Success Guarantee (postconditions):** Content is styled and formatted according to the appearance profile of either the Content Package or the institution, and the preferences of the user. Content is served to the user.

**Main Success Scenario (or Basic Flow):**

1. User requests content from institutional LMS.
2. LMS retrieves content from repository.
3. LMS retrieves appearance profile of the resource of the current organization in the Content Package.
4. LMS applies appearance profile of the resource to the content.
5. LMS applies appearance preference of the user to content.
6. LMS delivers content to user.

**Extensions (or Alternative Flows):**

- 3a. System overrides appearance profile of the resource with the appearance profile of the institution - i.e. branding.
- 4a. System fails in applying institution's appearance profile to content:
  1. System logs exception and/or notifies administrator.
  2. System delivers content to user 'as is'.

**Special Requirements:**

- Standards-compliant browsers and libraries must transform and render the content and stylesheets.

**Technology and Data Variations List:**

- 3a. Appearance profile of the resource consists of a CSS and XSLT.
- 3b. The XSLT transforms the content into XHTML.
- 4a. The XSLT is applied first and then the CSS is inserted into the XHTML document.
- 4b. The user style preferences, as expressed by CSS, are inserted into the XHTML document.

**Frequency of Occurrence:** Could be nearly continuous.

**Open Issues:**

1. What is the original data model of the content?

**Use Case DAM2: Institutional Content Profiling**

**Primary Actor:** Institution

**Stakeholders and Interests:**

- Institution: Wants all endorsed content to meet a number of mandated requirements specified in its content profile.
- User: Wants to receive requested content in a fast, easy manner.

**Preconditions:** Institution has a content profile.

**Success Guarantee (postconditions):** Content is retrieved and delivered to the user meeting all the requirements mandated by the institution's content profile.

**Main Success Scenario (or Basic Flow):**

1. User searches for content using institution's LMS.
2. LMS retrieves primary content metadata from repository.
3. LMS retrieves institutional content profile.
4. LMS verifies primary content meets all requirements mandated in institutional content profile.
5. LMS delivers appropriate content to user.

**Extensions (or Alternative Flows):**

- 4a. Primary content fails a requirement mandated in institutional content profile:
  1. LMS retrieves alternative to primary content metadata from repository.
    - 1a. Primary content has no (more) alternative content.
      1. LMS returns *'No results found'* to the user.
  2. LMS verifies alternative content meets requirements mandated in institutional content profile.
    - 2a. Alternative content fails requirements mandated in institutional content profile:

1. Go back to step 4a.1.

#### Special Requirements:

- Language internationalization must be supported.

#### Technology and Data Variations List:

- 2. Institutional content profile includes an EARL statement listing requirements of endorsed content - e.g. specifications, levels of conformance, platforms, etc.
- 3. Primary content metadata includes an EARL statement.
- 4a.1. Alternative content metadata includes an EARL statement and an 'isEquivalentTo' relation pointing back to the primary content.

**Frequency of Occurrence:** Could be nearly continuous.

#### Open Issues:

1. What is the vocabulary of the EARL statements in order to allow for intelligent decisions to be made by the LMS?
2. How do we handle cases where *some* of the content profile requirements are met by the primary or alternative content, but not *all*?

## Background

### The Relationship Between the DAM, an Institution and its Users

Institution

From Webster's Revised Unabridged Dictionary (1913):

2. That which instituted or established; as:

(a) Established order, method, or custom; enactment; ordinance; permanent form of law or polity.

A Dynamic Appearance Model for the above definition would consist of functionality uniformly applied across all content of which that institution endorses or owns. This functionality would consist of styling and/or retrieving the correct content from which the institution's "established order, method, or custom" is best represented. Another term for this type of functionality is "branding."

But, an institution represents and provides services to many individuals. These individuals, much like institutions, have their own "established order, method, or custom" and in some cases this is not a purely aesthetic quality or taste, as in the case of an institution, but rather a need of the individual essential to their interaction with the. What is the relation of these two potentially conflicting methods with respect to the DAM?

### The ACCLIP

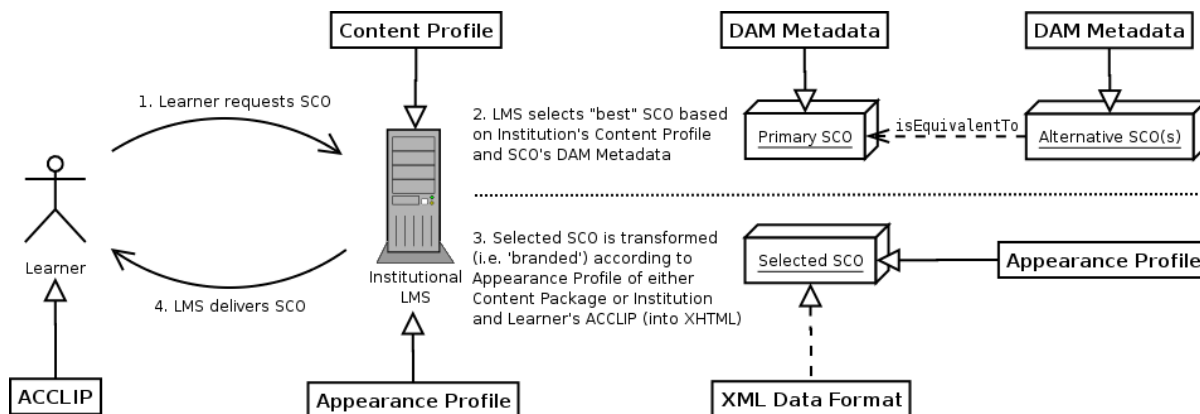
Previous work has been done on specifying what the characteristics of a learner's needs or *preferences* are in terms of providing equal access to all learners. This work was done in the IMS ACCLIP ( [Accessibility for Learner Information Package](#) ) specification. The DAM should keep the ACCLIP's work in perspective

at all times and, where possible, re-use techniques and not allow incompatibilities to arise between the two.

To what extent the DAM becomes a *special case* of the ACCLIP - i.e. one where an institution defines a set of ACCLIP preferences (as opposed to a user) - I'm not entirely sure of. The ACCLIP defines constructs which could achieve most of the binding requirements of the DAM - i.e. the Content and Appearance Profile - but it also defines a lot of other user-centric constructs which are outside the scope of the DAM requirements. Also, the ACCLIP is binded to XML in order to keep the data in a neutral format and hence, not define its intended environment (i.e. context-free) - e.g. it could be used to style a desktop UI, style a webpage, modify an application's functionality, transform displayed content, etc. The DAM, on the other hand, has a very specific enviroment: the SCORM API/Model. This, combined with the fact that it is to be applied at an institutional-level and that an institution has dominion over the LMS which is to implement the DAM, lead me to believe that a context-dependant solution is possible and perhaps even better. In other words, a solution which is context-dependant (like the DAM) and uses existing standards - e.g. CSS - rather than an abstract, context-free binding - e.g. the ACCLIP's `screenEnhancement` section - could be the best solution. Also, an abstract context-free solution essentially takes the 'lowest common denominator' approach in achieving its ends - i.e. it trades functionality for flexibility. Applying this solution to a context-dependant environment would result in no one receiving any benefits from the solution's flexibility (due to the content-dependant nature of the environment) and limit the system's funtionality (by virtue of the context-free solution).

Having the DAM endorse existing web standards in its specification would not preclude institutions (or learners) from using ACCLIP preferences. It would only require the system to first transform the ACCLIP preferences into the standards endorsed by the DAM and then processing could begin as initially specified.

### DAM Model Overview



### Appearance Profile

The Appearance Profile would be comprised of the following components:

1. an XML Stylesheet Language Transformation (XSLT)
2. a Cascading Style Sheet (CSS)

Both the XML and CSS stylesheets would be applied to a SCO prior to its delivery to the client. The XSLT would be applied first in order to transform the SCO into XHTML (if not already in that format --

[see below for discussion on SCO data formats](#) ). Then the CSS would be inserted into the XHTML document either directly or using an `import` statement.

The XSL transformation would handle the following styling / branding functionality:

- inserting a logo
- layout of header, footer, menubar
- displaying blocks of content in detail or in overview
- creating a table of contents

The CSS would handle the following styling / branding functionality for each layout section of the document - e.g. header, footer, menubar:

- foreground, background, highlight, link colour
- font size, face, family
- any additional styles

### **Content Profile and DAM Metadata**

The Content Profile and the DAM Metadata have a reflexive relationship in that all properties endorsed in the Content Profile must also be endorsed in the DAM Metadata. This is so that a *correct match* can be determined by the system at runtime. A *correct match* would be a SCO which:

1. is requested by the client and is either:
  1. the SCO itself - i.e. the primary SCO
  2. an alternative SCO equivalent to the primary
2. meets all the requirements of the Institution's Content Profile
  - In the case of when no SCO (primary or alternative) meets the requirements of the Institution's Content Profile, the "best" SCO will have to be determined
  - Note: an algorithm for determining the "best" SCO in this case could consist of querying the client as to what the user's preference is, returning the SCO that met the majority of the Content Profile's requirements, or simply returning the primary SCO

Some shared properties of the Content Profile and DAM Metadata relevant to styling / branding are as follows:

- locale and language
- compliance to specifications

For example, an institution could mandate that all content served by its LMS must conform to the [WCAG 1.0](#) . The institution would specify this in their Content Profile. The institution's LMS implementing the DAM would then know this constraint and so ensure that only SCO's marked up as conforming to WCAG 1.0 in their DAM Metadata would be made available to the users.

### **DAM-SCO XML Data Format**

In order to better facilitate the ability of the DAM to style and brand SCO content, a common XML data format should be endorsed by the stylesheets of the Appearance Profile and the SCO content. The following is a list of potential formats and the pros and cons associated with each:

- DocBook
  - Pros:
    - structured format for publishing documentation
    - longstanding standard
    - widely used, widely supported
    - modular, customizable DTD
  - Cons:
    - complicated - i.e. lots of tags
    - dynamic content is achieved through DocBook HTML Forms V1.1
      - new feature as of DocBook 4.1.2 -- not supported in Norman Walsh's DocBook XSL (yet)
- StyleBook
  - Pros:
    - subset of DocBook
  - Cons:
    - still complicated
    - only supports static documentation content - i.e. no forms allowed
- XDoc
  - Pros:
    - structured format for publishing websites
    - uses an XML/XHTML syntax
    - no global style information; only inline - e.g. the 'style' attribute
    - simple
  - Cons:
    - no published DTD or schema
- XHTML
  - Pros:
    - XML format of HTML
    - no transformation required for the content
    - widely supported in editors
  - Cons:
    - loosely structured data:
      - institution can not control the look and feel

- "bad HTML" - e.g. frames, popups, etc.
- possible global style information - i.e. CSS in head section applies to all elements in the document (even DAM generated ones)
- potential conflicts between variable, class or id names in XHTML and DAM-generated content
- XForms
  - Pros:
    - Adopted in XHTML 2.0
    - Reduces the amount of Javascript used within browsers
    - Direct browser support is unlikely for another couple of years, but an interim solution is server-side HTML translation
  - Cons:
- DITA
  - Pros:
    - Content 'reuse' through object-oriented design
    - Transformations and styles defined for the 'base' classes can be applied to any and all derivations
  - Cons:
    - New and unestablished
    - Ongoing development

## 1.2.3 DAM Test Cases

---

### DAM Test Cases

#### Introduction

This document lists test cases for the various transformations that are part of the DAM project.

For the present, this lists possible tests for each transformation individually. Ultimately, there needs to be a way of testing combinations of transformations.

The tests are listed by transformation. The names of the transformations and their priority order are based on the [Apr 5/04 face-to-face meeting notes](#), and subsequent teleconferences and emails. They are:

1. [Layout \(XSLT\)](#)
2. [Text CSS \(Text Rendering\)](#)
3. [Alternative Styling](#)

#### Role of the Acclip

The [IMS](#) defines the Learner Information Package schema, or [LIP](#). The purpose of the LIP is to provide a place to store information about learners and content creators. The [<accessForAll> branch](#) of the LIP is used to store display, control, and content preferences. This branch is referred to herein as the "Acclip".

The Acclip provides a way for users to declare their preferences in terms of displaying information, controlling their computer, and aspects of content. For example, it contains display preferences for font size, colour, volume, and speech rate; control preferences such as handedness, key repeat rate, and double click speed; and content preferences such as caption type, sign language alternatives, image, and textual alternatives, to name but a few.

It is important to note that these are only preferences. The Acclip does define how the preferences are realized. Consider the [content.learnerScaffold](#) preference as an example. This element allows users to declare that they want or require a dictionary, or a calculator, or a thesaurus, etc., or any combination thereof. This is tantamount to a set of "true/false" settings -- either the user wants, say, a calculator, or they do not. The Acclip does not specify where nor how the calculator is provided. It could be a physical device supplied by the instructor, or it could be a virtual calculator defined by JavaScript in a web browser. All the Acclip does is allow the user to declare that they want or need one.

The relevance of the Acclip to the dynamic appearance model is that, in some instances, Acclip preferences describe display properties. Font size is an example. For that matter, if dynamic appearance is extended to other modalities, such as audio, the Acclip can be used to define "display" properties of audio presentation; for example, synthesized speech preferences. In addition, Acclip alternative content

preferences can be used by a content delivery system to decide which instance of a number of equivalent objects to send to the user. In this regard, the Acclip is useful for initiating and constraining content transformations.

In summary, the Acclip is a specification that is already in use in terms of defining display parameters and content transformations. Where applicable, it is one of the inputs to the transformations, and is noted as such in what follows.

### **SCORM Issue**

There are aspects of the dynamic appearance model that intersect SCORM in terms of navigation and sequencing. Consider that a sequencing rule may result in the disabling of a "next" user interface button, or the visibility of some <resource>. In both cases, it is SCORM that is affecting the appearance of something on screen. The dynamic appearance model also modifies the enabled state and/or the visibility of parts of the display.

The dynamic appearance transformations are not intended to replace or otherwise compete with SCORM. Rather, the research investigates a subset of the elements listed in SCORM 1.3 to determine how appearance constraints can also play a role.

### **Layout (XSLT)**

When a user views a SCO using a Web based LMS, the screen is divided into a number of parts. The layout transformation determines the visual location of these parts.

The parts include a table of contents, tool bar, a navigation bar, header, footer, content, and logo. These are listed below in the order of priority.

Note that there are two sources or kinds of navigation, namely, SCO to SCO navigation, and page to page navigation *within* a SCO. The part that is used primarily for between-SCO navigation is the table of contents. The part for within-SCO navigation is the navigation bar, with the provision that any "next SCO" user interface element is disabled until reaching the last page of the current SCO. Similarly, "previous SCO" is disabled unless the user is viewing the first page of the current SCO.

#### Table of Contents

The table of contents is a "dynamic menu". Its purpose is to allow navigation among SCOs. It is derived from a content package organization item tree. Thus, this part of the screen is not derived from a SCO.

#### Tool Bar

The tool bar is a collection of useful "applications" that users can make use of while studying the material. Examples include a calculator and glossary. The tool bar is a "static menu". The tool bar is not derived from any SCO, but is provided by the LMS. Whether the user requires one is determined by their Acclip learner scaffold preferences. Whether these scaffold preferences are actually provided depends on their compatibility with the author or LMS' intended use of the SCO. For example, a user viewing a SCO whose type is "exam" is not allowed to use a dictionary or calculator. Such prohibition is keyed from the SCOs learning resource type meta data.

#### Navigation Bar

The navigation bar is primarily for *within* SCO navigation. It is derived from the major sections of a SCO. It changes as the user moves from SCO to SCO. There are two types of navigation: (1) page-to-page navigation within a SCO, and (2) SCO-to-SCO navigation. In page-to-page navigation, the "next/previous SCO" button should be disabled until the user reaches the last/first page. These are LMS settings.

#### Header and Footer

The header and footer are defined by the system and provided by the LMS. Thus, the header and footer are not part of a SCO.

#### Content

This is the content of the SCO itself. Think of it as the text surrounded by the table of contents, tool bar, navigation bar, header, and footer. The pre-transformed data model of the content will be DocBook v4.2.

#### Logo

A logo is a small image that designates the educational provider, or a sponsor, and so on. A logo may be defined in two places: (1) LMS configuration or, (2) in the SCO content. In the former case, the logo can be displayed in an LMS section of the page or, if communicated via an API, in the content part. In the latter case, the logo can either be embedded inside of the content itself or defined outside of the content and inserted into the content at runtime. The location of the logo can be determined through LMS settings and user preferences.

The intention is to separate screen presentation from content. As such, priority should be given to the header, footer, table of contents, tool bar, and navigation bar; and assume that the content will be transformed to fit inside the content area.

Each of these parts has properties that are relevant to a layout transformation:

- Visibility. Note that SCO *content* visibility is an exception. Content visibility is not specified by dynamic appearance, but by sequencing rules.
- Enabled.
- Location on screen.
- Active part (what happens to an element when the presentation device changes). This is of low priority and will not be addressed in this phase of the project. It is left here for future reference with regard to screen sizes, resolutions, etc.

### Layout Test Case

Given the parts described above, then the inputs to a test case consist of:

1. Content package manifest organization from which to define a table of contents.
2. Tool bar preferences derived from the Acclip, author or LMS restrictions, and SCO Learner Resource Type Meta data. These define the presence of the tool bar. The LMS provides the actual content.
3. DocBook markup that clearly designates sections/pages of the SCO, embedded logo, and pages.
4. Other clearly defined logo sources.
5. Multiple sets of rules, in plain English, or mock up graphics, that indicate where and how these parts

should appear.

The tests succeed when the XSLT transformations appropriately positions, hides, shows, enables, and/or disables the parts according to the English description and mock ups. The number of properties, and their combination implies a large number of test cases.

### **SCO Requirements:**

The DocBook markup needs to clearly designate content groupings to be used for the navigation bar and author defined logos. In addition, descriptions and/or mock ups are needed to show what the results of layout transformation is supposed to be. An assumption is that one DocBook file represents one SCO. An ongoing research is the relationship between the DocBook XSLT and the LMS-controlled user interface - e.g. frames vs. no frames, how to disable scaffolds when necessary, etc.

### **Text CSS (Text Rendering)**

Simply put: does the rendered document obey the CSS rules that were supposed to be applied to it? If yes, then it passes the test, and fails otherwise. Three test cases are briefly described below.

#### **What Content can a CSS be Applied To?**

Consider the parts noted above in the Layout Transformation section. If their "ultimate" data model is (x)html, then their rendering can be modified by CSS. "Ultimate" is that which is delivered to the user. For example, while a SCO resides in the repository in DocBook format, it will be transformed into (x)html before being delivered to the user's browser. It's that (x)html to which the CSS is applied. This, of course, is following the processing model as defined in "[Dynamic Appearance Model - Extending SCORM](#)".

Regardless, the proposal on the table is that CSS is applicable to all of the parts listed in the layout transformation section above, with the possible exception of logos.

#### **Text CSS Test Case: Branding CSS**

The test case is whether the institutional branding, as expressed in the LMS configuration, are realized. In this case, none of the screen elements (table of contents, navigation bar, etc.) have any inherent text rendering styles. Style information is based solely on the brand. The test is whether styles triggered by the brand are realized when delivered to the user.

#### **SCO Requirement:**

For testing the branding CSS on its own, the SCO must not contain nor link in any CSS styles.

Ongoing research will be done to determine the location of the branding CSS - i.e. is it at the LMS or in the Content Package? The branding CSS should not, however, be contained in nor linked via the XSLT.

#### **Text CSS Test Case: Acclip CSS**

The test case is whether the user's preferences, as expressed by their Acclip, are realized. In this case,

none of the screen elements (table of contents, navigation bar, etc.) have any inherent text rendering styles. Style information is based solely on a user's Acclip. The test is whether styles triggered by the Acclip are realized when delivered to the user.

**SCO Requirement:**

For testing Acclip triggered CSS on its own, the SCO must not contain nor link in any CSS styles. Likewise, the content package manifest organization and tool bar markup also do not import or otherwise include CSS styling.

**Text CSS Test Case: Branding CSS + Acclip CSS**

The test case is whether the institutional branding, as expressed in the LMS configuration, combined with the user's preferences, as expressed by their Acclip, are realized. Again, for this test case, none of the screen elements (table of contents, navigation bar, etc.) have any inherent text rendering styles. Style information is based solely on a user's Acclip. The test is whether styles triggered by the branding plus acclip combination are realized when delivered to the user.

This test case could be extended to a suite of tests of "canonical" branding or Acclip preferences; for example "high contrast", "large text", "colour blind", and so on. Note that these three may or may not be appropriate; they are listed here as examples to make the point.

**SCO Requirement:**

For testing branding + acclip triggered CSS on its own, the SCO must not contain nor link in any CSS styles. Likewise, the content package manifest organization and tool bar markup also do not import or otherwise include CSS styling, except, perhaps the branding CSS -- ongoing research will be done to determine the location of the branding CSS - i.e. is it at the LMS or in the Content Package?

**Text CSS Test Case: Branding CSS + Acclip CSS + Author/Multiple CSS**

In the past, this has been referred to as "cascading Acclips". It amounts to using styling rules from a number of sources and applying them at once to the SCO, and other parts of the screen. The test succeeds when the various CSS rules cascade appropriately.

**SCO Requirement:**

In this case, the DocBook markup, content package manifest, and tool bar source is allowed to reference one or more sets of "author-defined" CSS rules. By "author-defined" is meant any CSS rules *not* defined by branding nor acclip.

LMS configuration properties should decide the priorities of the CSS files and hence, the cascade order between the LMS, Acclip, and "author-defined" CSS. In other words, the cascade is *not* left up to the browser, but is done by the transformer and LMS prior to delivery to the browser. Ongoing research will be held to determine what attributes / rules the three CSS files are allowed to set and whether this is a result of their priority or rather innately determined by their type. Also, the LMS must pass its configuration properties to the transformer as one of the latter's inputs.

## Alternative Styling

Alternative Styling is concerned with these three issues:

- To what degree can content be styled in order to meet the institutional branding and/or learner preferences?
- What content properties are required in order to achieve effective styling?
- What does a system do if content cannot be styled properly?

In this instance, the concentration of effort is on the requirements for styling the content and less on content alternatives. There is a relationship with the above Layout and Text CSS sections, but it is at a higher level, namely, the level of a SCORM Styling API. Research should focus on what a styling environment is, and what the necessary factors are, in order to make styling as a whole work and, also, what to do when these factors are not present.

The meeting notes describe this transformation as follows: "The transformer examines a list of alternative SCOs and their accompanying meta data and selects the one that most closely matches a user's preferences".

There are two possible scenarios here, and two test cases for each scenario. The two scenarios are whether a specific SCO can be styled or not, and within each scenario, exact match vs. nearest match test cases.

### Scenario 1 - SCO cannot be styled

In this scenario, the SCO selected by the user cannot be styled by the transformer(s). In a sense, the style is fixed. Reasons include binary SCOs, legacy SCOs, and proprietary SCOs that are dependent on a specific client.

This case is handled by simply providing the SCO "as is", i.e. no attempt is made at styling the SCO. If this is inconsistent with the user's Acclip preferences, the LMS may warn the user that the content he/she is about to view may not meet their requirements, and query the user as whether or not to continue. The usage attribute on Acclip preferences can provide the LMS as to the importance of specific user preferences, and whether to display such an alert.

A possible solution for handling fixed-style SCOs is for the system to use meta data to find equivalent content that can be styled. This solution is of lower priority for this project, and will be addressed time permitting.

### Alternative Styling Test Case: Exact Match/Nearest Match

Under the scenario that the SCO cannot be styled, the case of exact match vs. nearest match is moot. The test case succeeds when the original SCO is delivered as is, with an alert keyed to the user's Acclip preferences.

### SCO Requirement:

The SCO should be constructed such that it cannot be styled.

**Scenario 2 - SCO can be styled**

In this scenario, the styling requirements can be realized on the SCO. The system should automatically style the content in order to meet layout specifications, institutional branding, user preferences, and/or author-defined styles.

**Alternative Styling Test Case: Exact match**

For a base line test, there should be a SCO that admits of multiple alternative styles that match the branding, user, and author preferences. The transformation succeeds when there is an exact match between the SCO delivered to the user, and those preferences.

**SCO Requirement:**

The SCO must be constructed such that its styling is as plastic as possible. This is achieved by separating the style information from its content.

**Alternative Styling Test Case: Nearest Match**

When a branding or user preference does not exactly match any alternative style, the situation reverts to the first scenario. In a sense, the SCO is partially fixed with respect to its style. In that case, the parts of the SCO whose style are plastic are styled appropriately, while the fixed aspects are left as is.

**SCO Requirement:**

The SCO must be constructed such that its styling is plastic for the most part, but will contain parts that are fixed.

## 1.2.4 Extending SCORM

---

### Associating a SCO with its Data Model

For a transformation to properly resolve itself, it must be performed on the correct data model - that is, the data model for which the transformation was written against and is, consequently, dependent on. Knowing the data model is required since an XSL style sheet assumes a specific kind of document as input. Here, "kind" refers to the schema or DTD for that document. For example, if the style sheet is to transform a DocBook document, then it presupposes that the input is an instance document conforming to the DocBook schema.

For a non-(X)HTML SCO to be properly transformed, its data model must be communicated to the corresponding LMS' transformation manager. For a transformed XHTML SCO or non-transformed (X)HTML SCO to be properly styled, the endorsed style of the SCOs organization must be known as well as any additional style information specific to the SCO. The following is a discussion on various techniques for communicating data models and style information of SCOs within the context of a Content Package in regard to the proposed DAM/SCORM model.

#### Note on vocabulary

The following is a list of common terms used throughout this document with a brief explanation of each:

Data Model

The internal structure of a document. Type, format.

Style

The purely aesthetic qualities of a document - e.g. colours, fonts, sizes.

Appearance

Both the data model restructuring and styling of a document.

#### File extension / MIME Media Type

A standard way web servers communicate the data model of a requested piece of content to the user agent is to associate the requested content's file extension with a MIME Media Type. The mapping between file extension and MIME Media Type is a series of file extension(s) / MIME Media Type properties set in the web server's configuration. Note: Magic numbers at the beginning of the content file might also be used to determine the MIME Media Type.

From a DAM/SCORM perspective, this technique is insufficient because different data models, including ones proposed in the [XML Data Format section](#), share common file extensions - e.g. '.xml' - and would, as a result of using this technique, have identical and, very likely, generic MIME Media Types - e.g. application/octet-stream. As a result this technique can not effectively communicate different data models which share common file extensions.

### Technical.Format Metadata of a Resource

First, a brief review of the structure of a `<resource>` element within the content package manifest is in order.

The `<resource>` tag itself is used to reference the primary unit of content. It picks out a SCO or asset. More importantly, a `<resource>` has an associated metadata section that is used to describe that resource/SCO/asset.

The `<resource>` element contains a list of `<file>` elements, each of which is a supporting asset for the primary `<resource>`. For example, the SCO may contain or reference images, applets, and other types of objects. Each of these objects is listed as a `<file>` within the `<resource>` section. And, again, each individual `<file>` has an associated metadata section.

With this structure in mind, the `technical.format` metadata element communicates the technical datatype(s) of all the `<resource>` and `<file>` elements used in the makeup of the SCORM Content Model Component. The data type of this element is a MIME Media Type based on IANA registration. Sections 4.2.5 and 4.2.5.1 of the SCORM Content Aggregation Model (Version 1.3 Working Draft1) state that in the Content Aggregation SCORM Application Profile, the multiplicity of the `<technical>` and `<format>` elements are both '1 or More' [1]. This guarantees that if a `<resource>` has metadata associated with it, the metadata must contain all MIME Media Types for the content in that resource.

In short, the data model required for a transformation of a SCO or asset can be declared in the content package manifest, by adding a single `technical.format` metadatum to each `<resource>` and `<file>` element.

From a DAM/SCORM perspective, this technique is sufficient if MIME Media Types are endorsed by SCORM metadata for all DAM-enabled data models. Currently, not all data formats discussed in the [XML Data Format section](#) have registered MIME Media Types with the IANA. Fortunately, the MIME RFCs allow for declarations of non-registered Media Types through an extension mechanism which can be used for internal or defacto data models [2]. As a result, it is possible to define non-standard MIME Media Types for all DAM-enabled data models and for this information to be communicated through `<resource>/<file>` `technical.format` metadata. Furthermore, the structure of the `<resource>`s element accurately reflects the mandatory communication of a SCO's data model with regard to the DAM/SCORM proposed model, as well as the possibility for a `<resource>` to contain multiple transformable data models.

### Extensions to Content Packaging

The IMS Content Packaging schema allows for attribute extensions in the `<resource>` element so long as they are from an external namespace and are valid within that namespace - i.e. 'strict' extensions. The MIME Media Type of a `<resource>` could be communicated through an attribute extension.

From a DAM/SCORM perspective, this technique is sufficient but duplicates the `technical.format` information in `<resource>` metadata. Also, the `<resource>` structure allows for multiple `technical.format`'s to be defined, whereas an attribute only allows for, at most, 1 MIME Media Type. This would not support the case where a SCO has multiple transformable data models.

## Associating a SCO with its Educational Type

Outside of a SCO's data model, its content can be described as portraying an educational theme or type. Some examples of this would be an exercise, simulation, questionnaire, and diagram. Communicating this information to a LMS would give it further potential to personalize and style a SCO.

### Educational.LearningResourceType Metadata of a Resource

In SCORM metadata the `<educational.learningResourceType>` element represents the specific kind of the SCORM Content Model Component. Sections 4.2.6 and 4.2.6.2 of the SCORM Content Aggregation Model (Version 1.3 Working Draft1) state that in the Content Aggregation SCORM Application Profile, the valid set of tokens defined by IEEE is:

- exercise
- simulation
- questionnaire
- diagram
- figure
- graph
- index
- slide
- table
- narrative text
- exam
- experiment
- problem statement
- self assessment
- lecture

and that the multiplicity of `<educational>` and `<learningResourceType>` is both '0 or More' [1].

From a DAM/SCORM perspective, this technique is sufficient because the `<educational.learningResourceType>` metadatum effectively communicates the educational type of the SCO it describes. Also, the optionality of the SCOs educational type within the DAM/SCORM proposed model is reflected through the `<educational.learningResourceType>`s '0 or More' multiplicity.

### Extensions to Content Packaging

The IMS Content Packaging schema allows for attribute extensions in a `<resource>` so long as they are from an external namespace and are valid within that namespace - i.e. 'strict' extensions. The educational type of a `<resource>` could be communicated through an attribute extension.

From a DAM/SCORM perspective, this technique is sufficient but duplicates the

<educational.learningResourceType> information in <resource> metadata. Furthermore, a <resource> is allowed '0 or More' <educational.learningResourceType> elements in its metadata, whereas a single attribute extension only allows for, at most, 1 educational type to be expressed for any given <resource> - i.e. a single attribute extension can not associate multiple educational types to a single <resource>.

## Declaring an Appearance for a Data Model and Educational Type

### The Relationship of Transforming and Styling

The two proposed technologies the DAM/SCORM employs in order to achieve a dynamic appearance model are XSLT and CSS. The XSLT is responsible for transforming content subscribing to an endorsed data model into XHTML. CSS is applied to XHTML in order to style the content. This is an ordered, two-step process with the XSL transformation being applied first and then the CSS being applied to the XHTML. Keeping the usage of XSLT and CSS separate - not only in terms of their application, but also in terms of their configuration - is ideal in order to keep the DAM/SCORM data loosely coupled. In other words, XSL transformations should not contain CSS style rules, but rather should be responsible for *structuring* the document only. The CSS should be responsible for *styling* a structured document only.

A transformed document should:

1. be XHTML
2. share a common structure with all the other XSL transformed documents

The structure should consist of standard, composite sections which are *classified* according to their *structural role* in the page - e.g. header, footer, sidebar, etc. CSS files, reflectively, should only style classes of elements based on these standard, composite sections and not style classes of elements based on their desired *display properties* - e.g. big, small, left, right etc. In this way the resulting content can, regardless of its initial data model, share a common structural Look & Feel across all <resource>s in a content package and, as a result, the CSS files are able to style against a common document structure and, henceforth, be shared across appearances.

### XML Binding

In order to achieve the proposed functionality, a content package will have to declare the supported *appearances* in its packaged content. The following is an example XML instance of an <appearance>:

```
<appearance identifier="basic" defaultCss="basic-default.css">
xmlns="http://dam.atrc.utoronto.ca/2004/05/DAM" <formatting mediaType="application/docbook"
xsl="basic-docbook.xsl"/> <formatting mediaType="application/x-qt+xml" xsl="basic-qt.xml"/>
<styling educationalType="simulation" css="basic-simulation.css"/> <styling
educationalType="exercise" css="basic-exercise.css"/> </appearance>
```

The DAM/SCORM should allow '0 or More' <appearance>s to be defined in a content package in order to accommodate all the supported appearances for its <resource>s. The <appearance>s should have unique identifiers in order to provide an <organization> the ability to reference/endorse an appearance and share it with other <organization>s. The <appearance> should define a default CSS file containing the common CSS rules to be applied to all its content. Also, a specific CSS file could be defined for each supported educational type in the <appearance>. This CSS file would contain the specific CSS rules for the educational type. This default CSS file should be applied before the educational type CSS file in order to allow the educational type CSS file to override the default CSS settings.

(Schema / Application Profile to be included shortly.)

### Associating an Organization with an Appearance

Alongside the technique to declare the Data Model and Educational Type of a SCO, an additional technique is required to declare what transformations and styles are supported by a content package for the various combinations of Data Models and Educational Types contained therein. As mentioned previously, an `<appearance>` should contain a unique identifier which can be referenced by an `<organization>` in order for an `<organization>` to endorse a supported `<appearance>`. The following is an example XML instance of an `<organization>` referencing a DAM appearance:

```
<manifest identifier="MAN1" xmlns:dam="http://dam.atrc.utoronto.ca/2004/05/DAM"> ...
<organization identifier="ORG1" dam:appearance="basic"> ... </organization> ... </manifest>
```

### Appearance Scenarios

A content package will declare its supported `<appearance>`s for each data model and educational type of the SCOs it packages. An LMS is allowed to override any of these settings with its own predefined appearances - e.g. institutional brandings. In other words, the `<appearance>`s defined within a content package should be viewed as styling recommendations - not as requirements.

A DAM/SCORM enabled content package should support all the content models and educational types of its contained `<file>`s. The metadata associated with a `<file>` is optional within a content package and, as a result, the data model and educational type are optionally declared for a SCO. (The educational type is furthermore optional within the DAM/SCORM proposed model outside of the metadata's optional presence). As a result, various cases exist pertaining to the presence or lack of DAM/SCORM information. The cases are outlined in the following:

SCO's `<file>` has a supported data model and educational type:

1. the data model's XSLT is applied to create an XHTML file
2. the default CSS file is applied to the XHTML file
3. the educational type's CSS file is applied to the XHTML file

SCO's `<file>` has no data model but does have a supported educational type:

1. assume the SCO has data model `text/html` or `application/xhtml+xml` - i.e. has already been transformed
2. the default CSS file is applied to the XHTML file
3. the educational type's CSS file is applied to the XHTML file

SCO's `<file>` has a supported data model but no educational type:

1. the data model's XSLT is applied to create an XHTML file
2. the default CSS file is applied to the XHTML file

SCO's `<file>` has no data model and no educational type:

1. assume the SCO has data model `text/html` or `application/xhtml+xml` - i.e. has already been transformed
2. the default CSS file is applied to the XHTML file

## Internationalization

TBD

## References

1. Advanced Distributed Learning SCORM Content Aggregation Model Version 1.3 Working Draft 1 ( <http://www.adlnet.org> )
2. Multipurpose Internet Mail Extensions [ RFC2045 , RFC2046 ] ( <http://www.isi.edu/in-notes/rfc2045.txt> , <http://www.isi.edu/in-notes/rfc2046.txt> )

## 1.2.5 DAM: Flow of Information

---

### DAM: Flow of Information

#### Basic View

The basic flow of information is captured in Figure 1, below: given a SCO and the details of how it should be transformed (the "Xform info" in the figure), the transformation engine engine modifies the SCO to produce a transformed version of it.

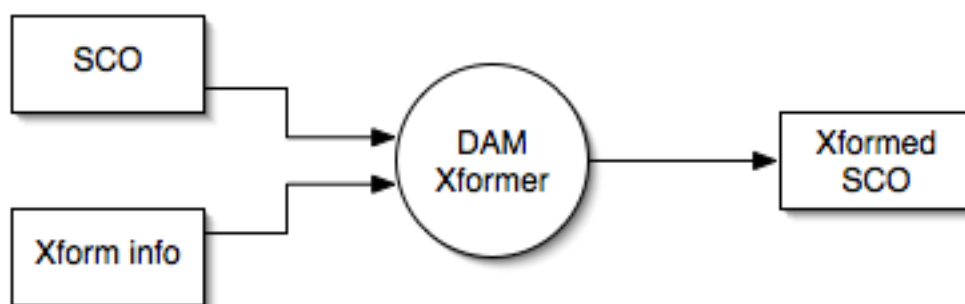


Figure 1. Basic Flow of Information in a DAM Transformation.

This is an intentionally oversimplified view of the transformation process, and is meant to provide a starting point. In terms of content density, figure 1 represents the most collapsed view of the process. What is needed is to expand each of the boxes to state explicitly what information is inside that box, and to expand the circles to specify what process(es) it represents. Another issue is the sequence of the transformations.

In what follows each transformation discussed in "[DAM Transformation Requirements](#)" is represented as an information flow diagram. There is some attempt to divest the transformation of its TILE influence, and describe the situation in a more general way. The approach is inspired by Jutta's comment (personal communication) that transformations should be thought of horizontally, and that they are not specifically for branding, or for different devices, or for accessibility, or for anything in particular, but are transformations that prove useful in multiple contexts. The attempt at generality is not completely successful, since trying to state what the transformation is without reference to at least one concrete implementation is difficult.

#### Text Rendering

All else being equal, it is fairly certain that the last transformation is the one that deals with font, sizes, colours, and so on. This is illustrated in Figure 2.

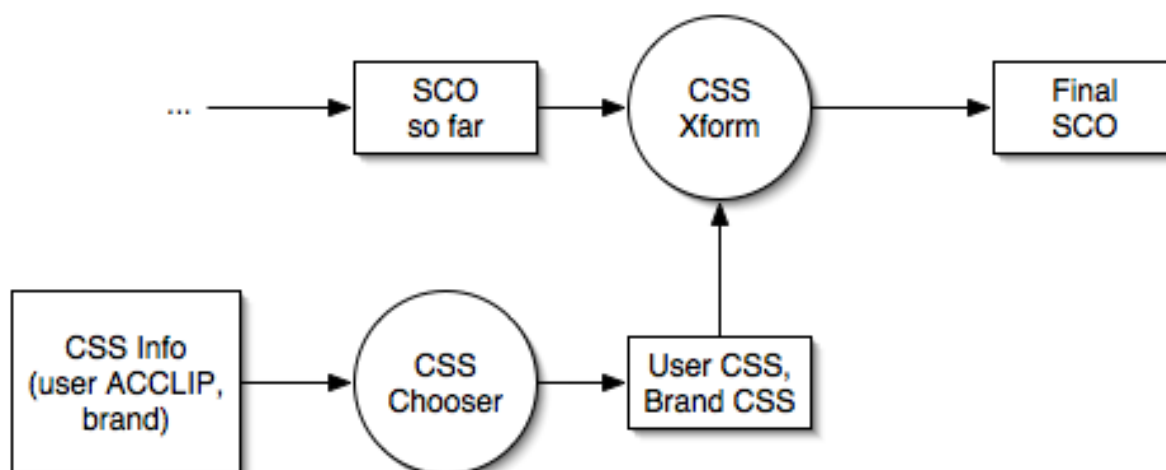


Figure 2. The Last Stage: Text Rendering Transformation.

Note that the actual rendering of the text takes place in the SCO viewer (the browser). All that the CSS Xform does is to inject the CSS information into the SCO before it is sent to the browser. The injection consists of the addition of <link> or a <style> element(s).

### TILE Method

" [TILE Text Rendering Transformation](#) " describes the TILE version of the text rendering transformation fully.

### Questions

1. Is this the locus of the layout transformation? The [requirements document](#) suggested that <div> and <table> elements will be used here.
  - This raises another issue. Are the SCO's going to be marked up in terms of major sections that partake in visual layout? That is, the major sections that can be moved about are encoded without regard to where they actually end up. Subsequent to that, the CSS is applied at the text rendering stage to render the layout according to the user's preferences and according to the brand.
  - Or, is there another transformation that applies some XSL transformation to insert the <div> and <table> elements for layout? If so, then that is another transformation that happens previous to this one.
2. The diagram follows the TILE assumption that the user's CSS is defined by their ACCLIP. It has been suggested that the ACCLIP could also be used for defining branding CSS. Some doubts have been raised against the latter in the [model document](#) . If not the ACCLIP (for branding), then what provides this information? Possibilities include:
  - An expanded ACCLIP.
  - Meta-data.
  - The adlnav namespace.
  - A new schema entirely.

### Layout

For the sake of completion, here is a diagram of the Layout transformation from the point of view that it is an XSL transformation that places sections of the document into visual locations.

Viewed this way, the layout transformation happens prior to the Text Rendering transformation.

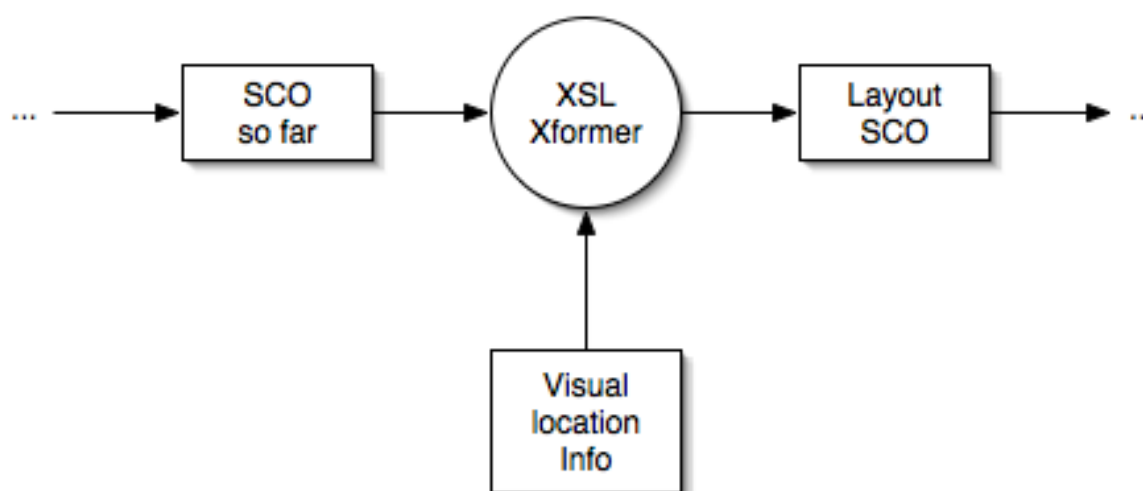


Figure 3. Layout Transformation.

### DOM Order

Figure 4 shows two possible interpretations of the document ordering transformation. Another possibility is that some combination of meta-data and XSL rules is used to create the re-ordered SCO.

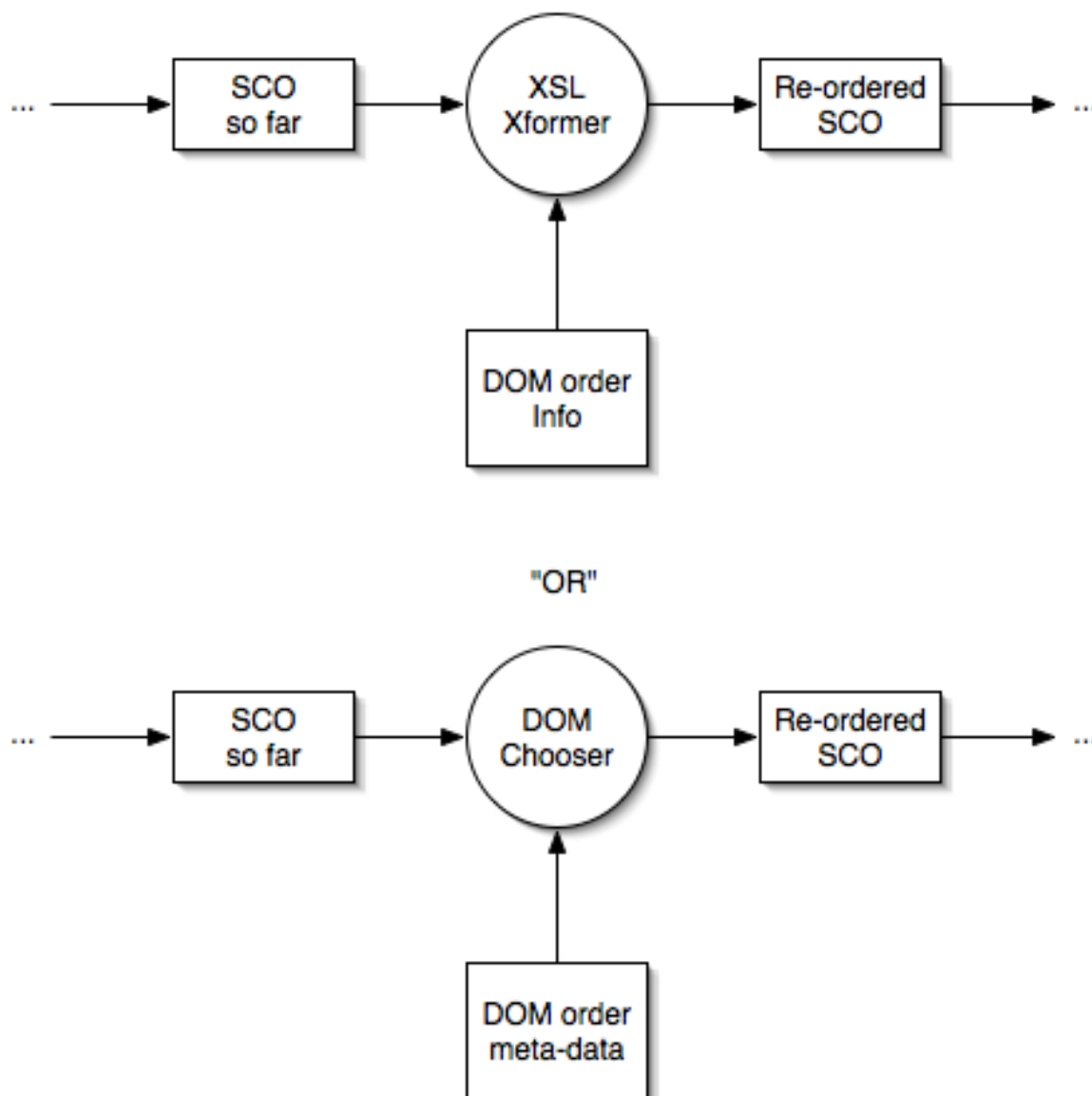


Figure 4. Two Views of the DOM Order Transformation

The top diagram in figure 4 is meant to show the idea that the document is reordered according to some XSL rules and an XSLT on the document. The bottom diagram shows the idea that meta-data could be used to choose among a set of "alternative" documents that differ from one another in terms of their ordering.

### Question

If it is the case that all of these alternative documents can be generated by some set of XSL rules, then these diagrams are logically equivalent, and the top method is preferable. Are there any reasons to suspect that's not the case?

### Exclusion (Navigation)

This is actually where the transformation is *not* a transformation on a SCO. Rather it is a navigation device that chooses which SCOs are visible. In TILE, the choice is based solely the user's preference; nothing else in the system affects that choice. This is the reason for the parenthetical "user pref" in the exclusion information box. For the DAM, it need not be this way; in fact, the exclusion information will also involve whatever branding requires for exclusion.

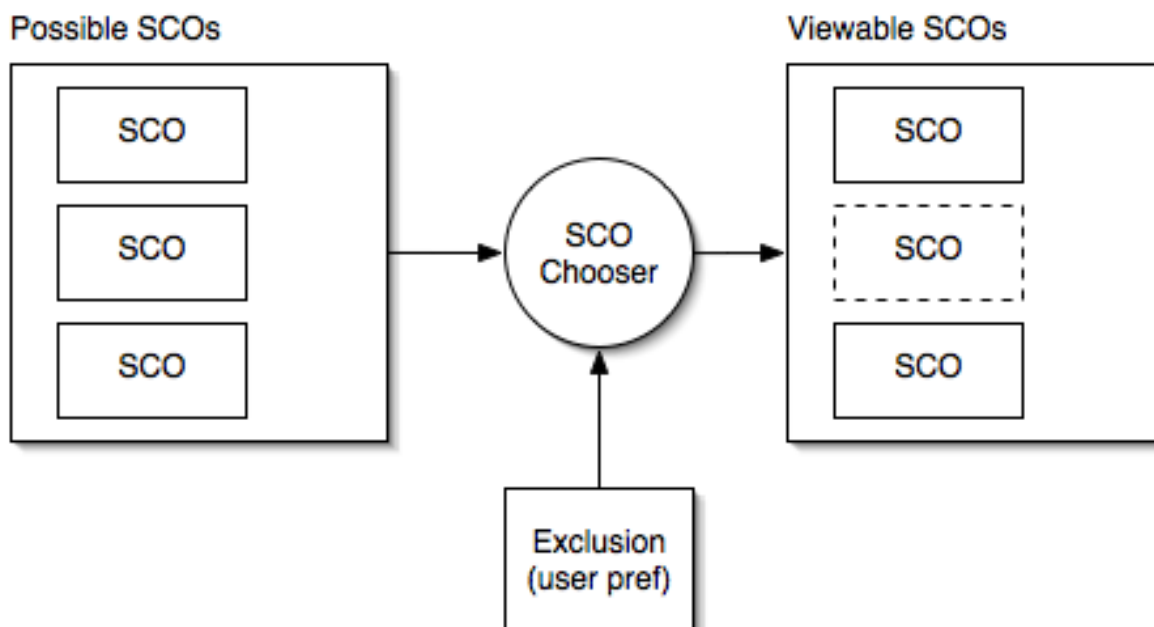


Figure 5. Topic Exclusion

### Questions

Is this a case where the branding might override the user? That is, if branding requires that a SCO be invisible, and the user desires to see it, who wins?

Since this is the user's choice in TILE, this transformation occurs live at the browser. The user can mark things in a table of contents view as excluded, and as this is done, the back end servlet makes a note as to which topics should not be automatically delivered by "next/previous" requests. As such, the transformation always takes place at the end of the process. Note that this is partly because no SCO is transformed at all at this point. Will this technique work in the case of the DAM?

### Alternate Content

### Content Density

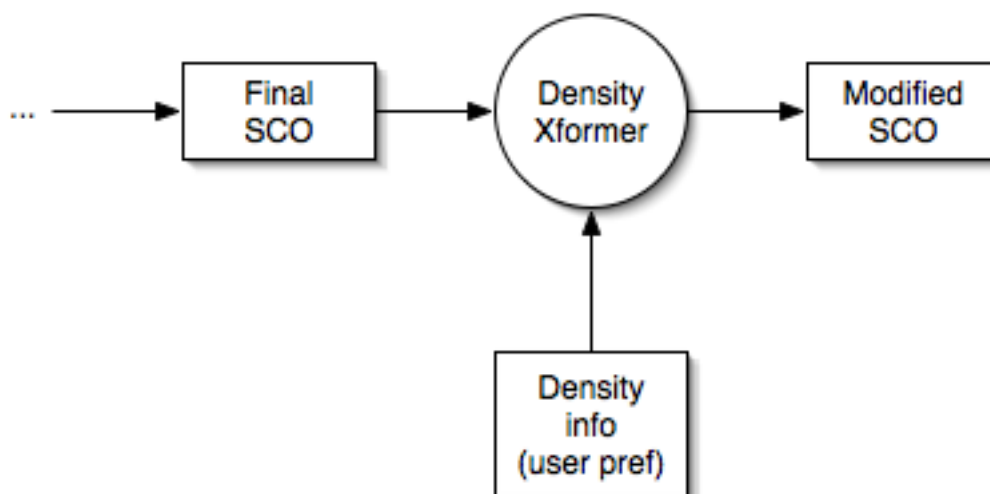


Figure 6. Density Transformation

Again, like topic exclusion, the transformation takes place on the client and is live. It is a toggle that the user can switch to collapse and expand content..

#### Content Views (Image vs. Text Intensive)

Figure 7 shows the content views transformation, and is based on the details given in the [transformations requirements](#) document.

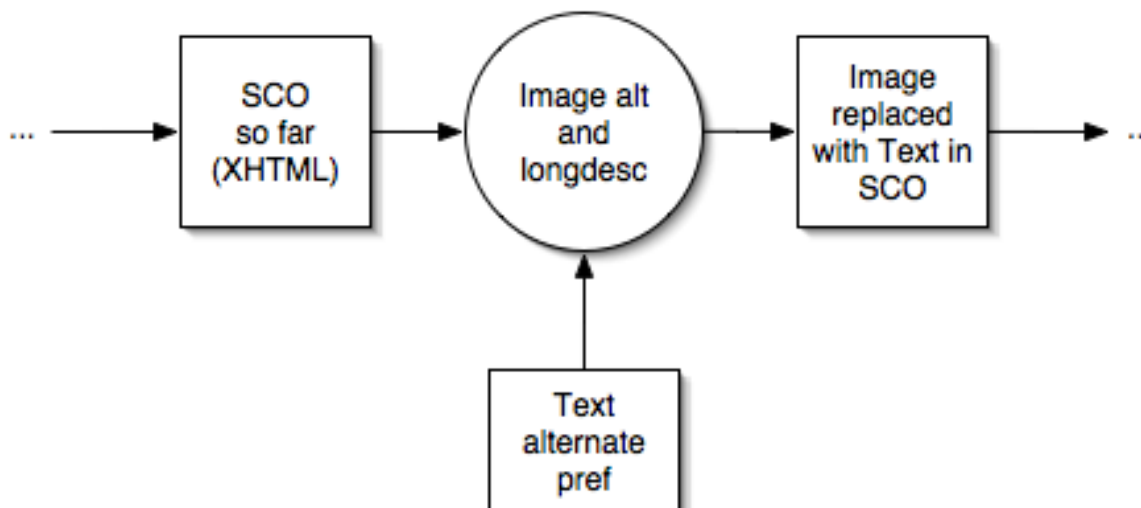


Figure 7. Image to Text Intensive Transformation.

This is the least general specification so far. The SCO is an (x)html document with image tags properly marked up with `alt` and `longdesc` attributes. The user preference for text intensive views of images drives the transformer to acquire the alt and longdesc text, and insert them as a paragraph in the (x)html.

#### Alternatives to X

Figure 8 shows an abstraction of the flow of information for all of the alternatives to X transformations. These include textual, graphic and, audio alternatives.

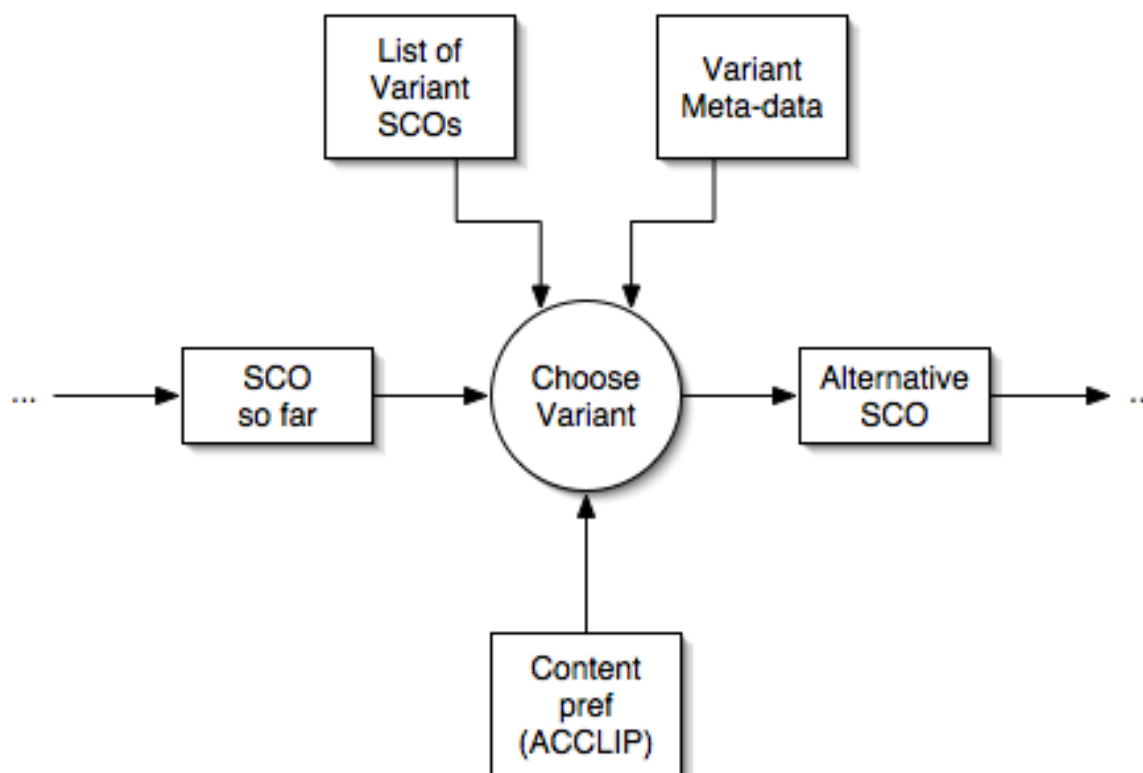


Figure 8. Alternative Content Transformation

The transformation's two triggering inputs are a list of alternative SCOs and meta-data on each of them defining the way that they are a variant of the input SCO. The transformer's job is to choose the variant SCO that matches the meta-data requirements, and the preference(s) that states which alternative is needed.

### Content Supplements

Not sure how to diagram this one...pending.

### Structure Presentation (Show/Hide TOC)

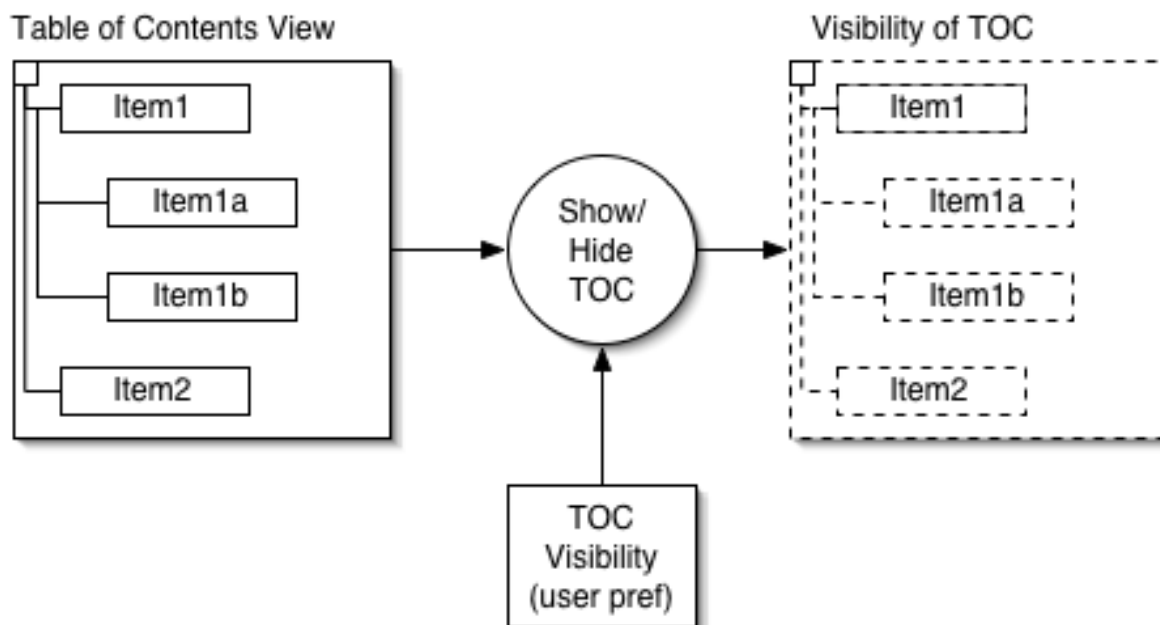


Figure 10. Show/Hide Table of Contents

Like the topic exclusion and content density transformations, the visibility of the table of contents is handled as a toggle. Its initial value depends on the user's ACCLIP preference, but thereafter the user can quickly switch the sense of the toggle. [\*\*Is this handled by a servlet, or JavaScript on the client?]

Technically, the table of contents is actually a view on the item tree in the content package manifest. It is not a list of SCOs as in the figure. However, this is a short cut for the figure.

### Structure Navigation (Depth vs. Breadth)

Figure 11 diagrams the depth-first vs. breadth-first navigation transform. Note, again, that this is not a transformation on an individual SCO, but occurs at a higher navigational level.

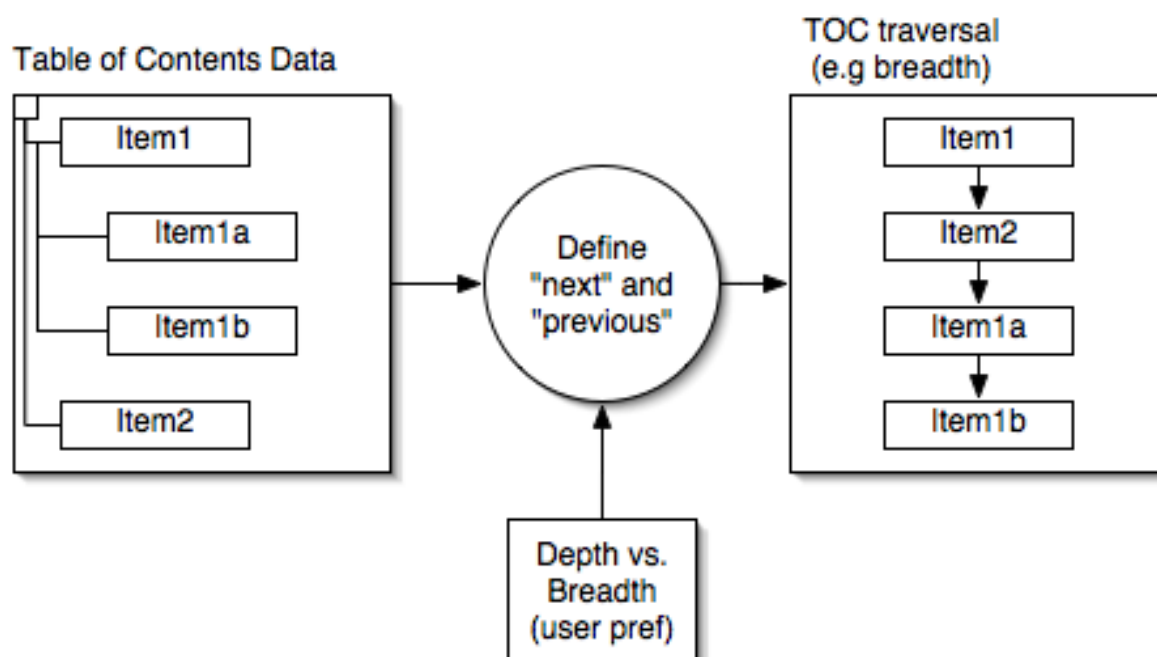


Figure 11. Table of Contents Traversal

Somewhere, there is a table of contents data structure that represents the topics/activities/objectives of the presentation as a whole. That table of contents is organized in a hierarchical fashion. In TILE, this is the item organization in the content package manifest.

The transformation consists of defining how the table of contents hierarchy is to be traversed. The diagram illustrates a breadth-first traversal of the hierarchy -- first the top level items are shown in sequence, followed by the items in the next level of the hierarchy. In comparison, a depth-first traversal of the same hierarchy would proceed:

Item1 -> Item1a -> Item1b -> Item2 ...

### General Issues

1. All of the above assumes that the transformations occur in isolation and in a linear fashion. That would be nice... But, the reality may be quite different; for example, the user preferences may preclude some transformation. There is a chance that some evaluation process needs to take into account and judge which way the transformation will go, if at all. Also, there is the possibility of feedback loops in that a transformation might influence a previous one. Note that TILE avoided this by performing each transformation in isolation and in a certain order.
2. There needs to be a break down of what is in each of the "info" boxes that influence a given transformation. These include:
  - User ACCLIP
  - Branding settings.
  - Meta-data.
  - EARL assertions about the feasibility of a transformation (?)

- Content package (manifest/item tree).
3. Are TILE transformations too specific to TILE to be used here without modification? How much of TILE should be used/pirated?

#### **Appendix - Transformation Order in TILE**

A goal is to place all of the above transformations into a sequence. By way of comparison, the following diagram shows the order of transformations in TILE. It differs from the above mini-diagrams in two ways. First, the relevance of the content package is emphasized, and second, the role of the ACCLIP is implicit as an input to each transformation.

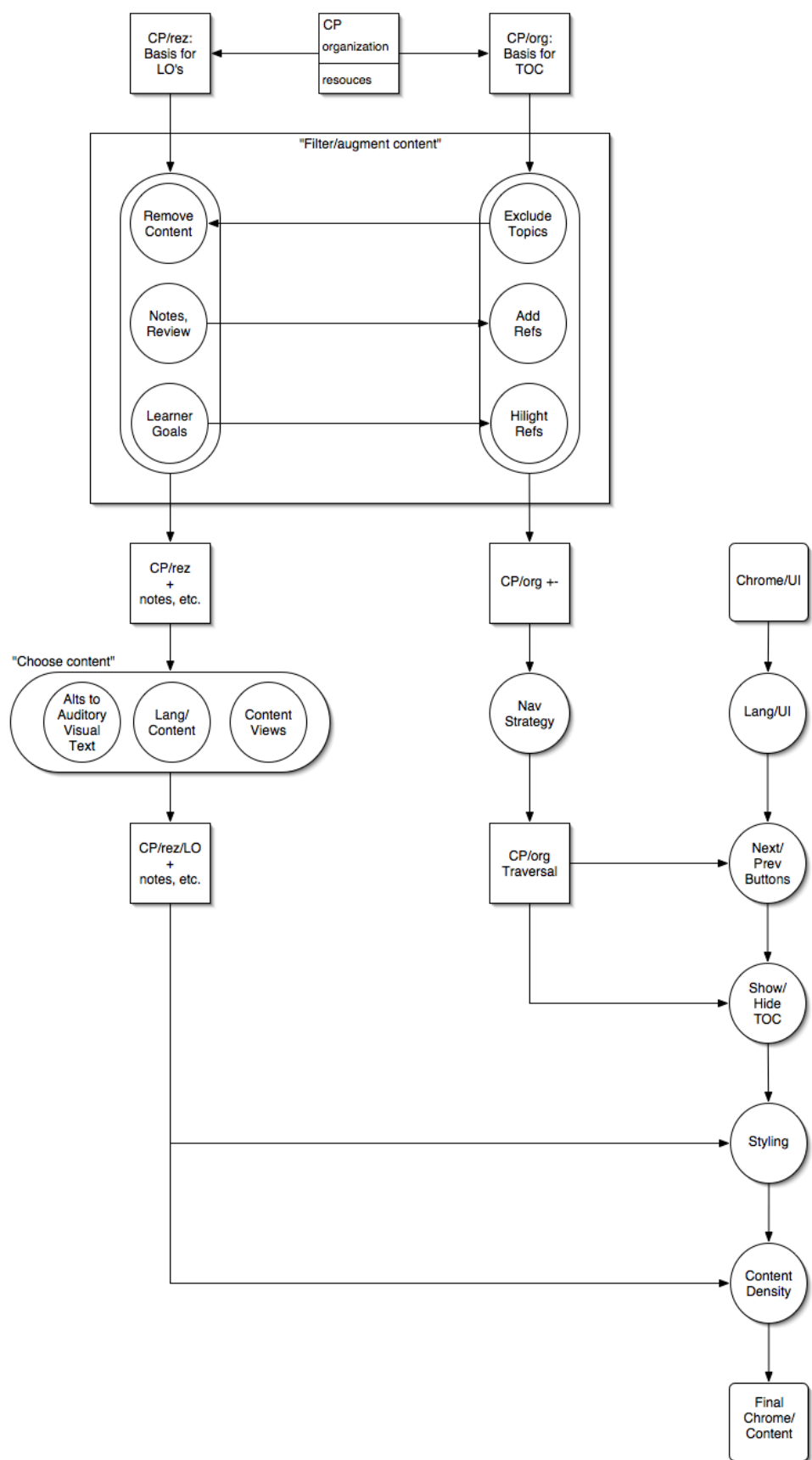


Figure 12. TILE Transformations.

In brief, content packages have two parts, namely, an organization of items and a list of resources. Some of the transformations are done based on the resources, and this is shown on the left. Other transformations are relative to the organization, and these are shown on the middle-right.

"Chrome" refers to user interface elements that are constant as different learning objects are presented to the user. Examples of the chrome are the next and previous buttons that are used to move from learning object to learning object. Some of the chrome is defined or at least influenced by a given transformation; the table of contents view is an example.

## 2.1 DAM Implementation Notes

---

### Notes

#### DAM in a Nutshell...

The essential issue concerning the DAM is how we achieve a model in which content can be branded by an LMS in order to achieve a consistent "Look and Feel" across all rendered SCO's, but also allow for SCO's to be sharable across different LMS platforms. The answer to this question is simple: separate *all* presentation from content. In this way only can a system achieve full control over the presentation of the content in order to achieve a consistent brand and L & F. The consequence of this is that the authors of the content have no say in the presentation of their content. This relationship is significantly different from what we experience today as users of the Internet. On the Internet, authors of content not only author the content information, but also have *full* control over its presentation - i.e. structure, colours, ordering, etc. For a complete dynamic appearance model to be achieved in which system or institutional branding is supported, all presentation will have to be removed from the content - including the author's presentation.

After all presentation has been removed from the content, what's left is the content ideally contained in a structured, semantic markup. The more structured and semantically defined the markup is in the content, the more control the system has in terms of presenting the content in a uniform fashion across rendered content. For this reason the test implementation of the DAM chose DocBook (and TEI perhaps, we should probably have two) as its two supported content models.

#### A Constellation of Terms

There are a number of concepts currently in use in the fields of e-learning, distance learning, learning objects, learning object repositories, and SCORM. These are *re-purpose*, *sharable*, *transformable*, and *accessible*:

- *Re-purpose*: Re-use the same content in different lectures, presentations, courses, and so on.
- *Sharable*: Allow content to display with any presentation system.
- *Transformable*: The ability to spawn different data formats without changing the essential character of the content.
- *Accessible*: The ability to peruse content using different means of presentation (e.g., a screen reader).

The above is not meant to define each term completely, but to highlight the main idea behind each, and to suggest that they are interrelated. First, note that these terms all apply to content or documents -- it is content that is { re-purposed, shared, transformed, accessed }.

Secondly, to the extent that content is *transformable*, it will work in a number of environments. That is, it can be *shared* among different presentation systems such as desktop web browsers, hand held devices, and pdf viewers, to name a few. Furthermore, it can be *re-used* in different "lesson plans". Accessible content,

by definition, is transformed into some other form -- braille, synthetic speech, larger text, and so on. Simply put, being transformable is at the root of these notions of sharing, re-purposing, and re-displaying.

Assume that *transformability* is the ideal case for content; what follows? Such plasticity implies that the content is of a form that makes little commitment as to how it will be used or displayed, since the more it does define its appearance, the less likely it will transform properly for use in another context. In the extreme, content that is completely neutral with respect to its appearance permits maximum transformability.

### **Any Middleground?**

In the above "DAM in a Nutshell..." section a trade-off was presented between author control of content presentation and system ability to brand content to a common L & F. Is there any middleground between these two extremes? That is, can the system and author agree at any level concerning the presentation of the content? And if so, how does this affect the DAM in terms of its ability to provide systems with a consistent branding model? What is the middleground, if any?

The criterion can be stated thus: the author is allowed to define presentation insofar as it does not interfere with potential transformations. For example, if the content is to be displayed on a hand held device with a small screen, but the author's styling force a large display, then it cannot be transformed for that case.

However, stating the criterion is one thing. Giving specifics for where and when it is applied is more difficult.

### **Special Cases**

Using the above model, a number of cases arise which need to be handled:

- What happens when a system does not support the semantic markup of the content?
  - Have the author provide a style?
  - Ignore the content?
  - Display the bare content?
- Should the DAM recommend a basic list of endorsed semantic markups?

## **Implementation Requirements**

### **DAM Manager**

- Reference to institutional XSLT file.
- Content type outputted by XSLT file.
- Reference to institutional CSS file.

### **XSL Transformation**

To be defined / referenced in XSL file:

- Institutional information:
  - name (translated)
  - url
  - logo
  - nag email address
- Reference to Content Package for TOC generation.
  - Question: what is the relationship between the TOC and the navigation bar?
- Namespace, Schema or DTD of supported data model(s).
- Supported languages(?)

Norman Walsh's DocBook XSL contain the following template classes:

- user.head.content
- user.header.content
- user.header.navigation
- user.footer.content
- user.footer.navigation

These could be edited in order to add institutional and user brandings.

### **CSS Styling**

- Institutional CSS file could be one of:
  1. URL which is linked in.
  2. File (or URL) whose content is inserted into the document.
  3. Default CSS file containing tokens and properties file defining token values which are then substituted (inside the CSS) - i.e. Maven XDoc plugin.
  4. CSS created from an XML markup - e.g. the ACCLIP
- Learner CSS file is:
  - CSS created from the learner's ACCLIP.